

Towards HPC-ABDS: An Initial High-Performance BigData Stack

Judy Qiu*, School of Informatics and Computing, Indiana University, Bloomington, USA

Shantenu Jha*, RADICAL, Rutgers University, Piscataway, USA

Andre Luckow, RADICAL, Rutgers University, Piscataway, USA

Geoffrey C.Fox, School of Informatics and Computing, Indiana University, Bloomington, USA

* Contributed Equally

Many scientific problems depend on the ability to analyze and compute on large amounts of data. This analysis often does not scale well, its effectiveness hampered by the increasing volume, variety and rate of change (velocity) of big data. There is a need to integrate features of traditional high-performance computing, such as scientific libraries, communication and resource management middleware, with the rich set of capabilities found in the commercial Big Data ecosystem, resulting in an integrated system generically called high-performance big data system (HPBDS). Our proposed preliminary implementation of the HPBDS – including many important software systems such as Hadoop available from the Apache open source community and thus referred to as High-Performance Computing-Big Data Stack (HPC-ABDS) – has two fundamental building blocks: (i) Middleware for Data-Intensive Analytics and Science (MIDAS) that will enable scalable applications with the performance of High Performance Computing (HPC) and the rich functionality of the commodity Apache Big Data Stack. (ii) The second building block will design and implement a set of cross-cutting high-performance data-analysis libraries SPIDAL (Scalable Parallel Interoperable Data Analytics Library), which will support new programming and execution models for data-intensive analysis in a wide range of science and engineering applications. These libraries will be implemented to be scalable and interoperable across a range of computing systems including clouds, clusters and supercomputers. The project libraries will have the same beneficial impact on data analytics that scientific libraries such as PETSc, MPI and SCALAPACK have had for supercomputer simulations. In this paper, we study many Big Data applications from a variety of research and commercial areas and suggest a set of characteristic features and possible kernel benchmarks that stress those features for data analytics. We draw conclusions for the hardware and software architectures that are suggested by this analysis.

General Terms: Big Data, HPC, Apache Hadoop

ACM Reference Format:

ACM 1, 1, Article 1 (August 2014), 23 pages.

DOI: <http://dx.doi.org/10.1145/0000000.0000000>

1. INTRODUCTION

The growing importance of data is evident in many fields including physical and biological sciences, and the ability to derive insight and knowledge from increasing volumes of complex data points is essential for advanced analytics. Analytics needs to be able to utilize the full range of available infrastructure. However, the coupling between tools, analytic engines and infrastructure is often rigid, thus it is difficult to employ existing solutions for contemporary environments for which they were not natively or originally designed. Further, many tools were developed at a time when parallelism was not essential, while interoperability at multiple levels remains elusive.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2014 ACM 1539-9087/2014/08-ART1 \$15.00

DOI: <http://dx.doi.org/10.1145/0000000.0000000>

The importance of advanced analytics to derive insight and knowledge from increasing volumes of complex data will continue to grow. The enterprise community has made impressive gains and seem to have converged around the Apache stack. A distinctive feature is the existence of many implementations of the specific components of the Apache stack, providing sufficient richness in the tradeoff between performance and capability. In contrast, within the scientific computing community, progress has been reliant either on long-term foundational advances or short-term hardware fixes. In both domains, scalable yet general-purpose and broadly applicable solutions in the form of analytic libraries and abstractions are noticeable by their absence.

To remedy this major gap and proffer an integrated solution that brings the best of recent advances to the service of extreme-scale science requirements on current and future science production platforms, we are developing HPC-ABDS – a first implementation of a high-performance Big Data stack (HPBDS) that integrates the best of the Apache developments and HPC capabilities. HPC-ABDS will utilize and expose the integrated relative technical strengths of the two hitherto disjoint approaches and communities, yet it will focus on delivering these as production grade implementations that will bring the best-of-both to shared-infrastructure – such as NSF’s XSEDE, DOE’s leadership machine, OSG and other domain-specific infrastructure, as well as the software developments underway as part of the SI2 software program. HPC-ABDS will translate these applications characteristics, infrastructural requirements and existing capabilities into well-defined and implemented building blocks.

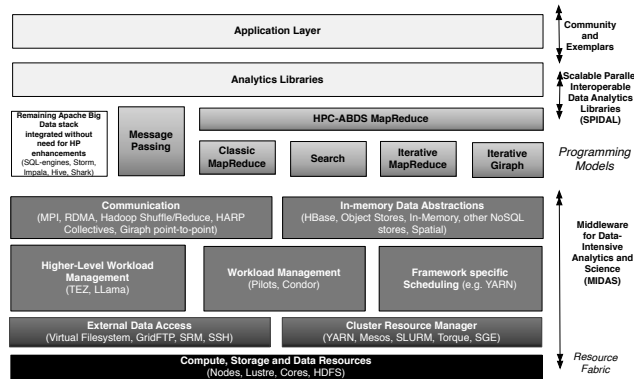


Fig. 1. Key components of integrated HPBDS stack. Many capabilities unaffected by integration are not shown explicitly

The key components of such an integrated platform are shown in Fig. 1. The aim of HPC-ABDS is to aim for the performance of HPC and the breadth and productivity of ABDS. The resultant integrated architecture is targeted at both production high-end computing platforms (such as leadership machines and XSEDE), as well as (commercial) cloud computing. As part of HPC-ABDS, we propose two fundamental building blocks, Middleware for Data-Intensive Analytics and Science (MIDAS) and the Scalable Parallel Interoperable Data Analytics Library (SPIDAL).

The high-performance community has prospered thanks to libraries like MPI, PETSc and SCALAPACK; SPIDAL brings this concept to data-intensive applications. SPIDAL Parallel Analytics Libraries will capture system abstractions and expose application requirements. MIDAS is the middleware upon which to build interoperable yet high-performance libraries. SPIDAL will enable interoperable high performance

data analytics and is based upon a careful analysis of architectures, tools and application characteristics/requirements. Although built from many existing components and capabilities, MIDAS is conceptualized and designed from first-principles to ensure achieving our productivity, interoperability and performance goals.

In earlier work [Jha et al. 2014b], we have discussed the need for merging the two “common” stacks. In addition to a qualitative motivation, Jha et al. [2014b] provided a quantitative analysis of the type of abstractions and support required to enable a successful hybrid stack. In this paper, we will move from a general motivation of the need for a hybrid approach to a discussion of the design philosophy & objectives in a specific implementation of HPC-ABDS, which serves as a first prototype towards a general purpose, interoperable high-performance Big Data stack to support analytics on high-end clusters, clouds and supercomputers.

2. SOURCES OF INFORMATION

Years of research have afforded us a plentiful if admittedly incomplete trove of information regarding Big Data applications. Here we selected 3 specific sources to utilize: a recent use case survey by NIST from Fall 2013 [NIST 2013a]; a survey of data intensive research applications by Jha et al. [2014a, 2013]; and a study of members of data analytics libraries including R [R Project 2012], Mahout [Apache Mahout 2012] and MLLib [MLLib 2014]. The following will examine these studies in greater depth.

In June 2013, NIST Big Data Public Working Group (NBD-PWG) began its run focusing on a set of working groups. These included Big Data Definitions, Big Data Technology Roadmap, Requirements, Reference Architectures White Paper Survey, Reference Architectures, Security and Privacy Requirements, Security and Privacy Reference Architectures and Taxonomies. 51 use cases were collected from a public call by the Requirements group and then analyzed on the basis of reference architecture requirements [NIST 2013b]. From this we extrapolate shared patterns and features we deem essential to incorporate in the future development of Big Data hardware and software. Out of these 51 use cases we formed nine areas of interest (the number of associated use cases are in parentheses): Astronomy and Physics (5); Commercial (8), Defense (3); Deep Learning and Social Media (6); Earth, Environmental and Polar Science (10); The Ecosystem for Research (4); Energy (1); Government Operation (4); and Healthcare and Life Sciences (10).

It is evident that research applications formed the core for a majority of these use case, with smaller attention paid to commercial, defense and government operations. Further efforts by the Requirements working group which allowed experts to categorize each use case by 26 features:

Use case Actors/Stakeholders and their roles and responsibilities; Use case goals and description; Specification of current analysis covering compute system, storage, networking and software. Characteristics of use case Big Data with Data Source (distributed/centralized), Volume (size), Velocity (e.g. real time), Variety (multiple datasets, mashup), Variability (rate of change); so-called Big Data Science (collection, curation, analysis) with Veracity (Robustness Issues, semantics), Visualization, Data Quality (syntax), Data Types and Data Analytics. Also included were general observations regarding Big Data Specific Challenges (Gaps), Mobility issues, Security and Privacy Requirements, and those issues singled out for generalization of this use case.

To view the full list of responses in addition to a summary from the working group of applications, current status and futures and extracted requirements, one can consult [NIST 2013b]. All 51 responses are located in the Appendix along with 20 other NBD-PWG use cases that did not exhibit the 26 features previously mentioned. The 20 generally fall under the domain of enterprise data applications and security and privacy.

Table I. What is Parallelism for NIST Use Cases

General Class	Examples
People	Users (but see below) or Subjects of application and often both
Decision makers	Researchers or doctors (users of application)
Items	Experimental observations Contents of online store Images or Electronic Information nuggets EMR: Electronic Medical Records (often similar to people parallelism) Protein or Gene Sequences Material properties, Manufactured Object specifications, etc., in custom dataset
Modeled entities	Vehicles and people
Sensors	Internet of Things
Events	Detected anomalies in telescope, credit card or atmospheric data
Graph Nodes	RDF databases
Regular Nodes	Simple nodes as in a learning network
Information Units	Tweets, Blogs, Documents, Web Pages, etc. and characters/words in them
Files or data	To be backed up, moved or assigned metadata
Particles/cells/mesh points	Used in parallel simulations

2.1. Properties of the 51 NIST Use Cases

Fox and Luckow [2014] summarize the characteristics of the 51 use cases and introduce the concept of Big Data Ogres for describing and characterizing Big Data applications. Note that Big Data and parallel programming are intrinsically linked as any Big Data analysis is inevitably processed in parallel. Parallel computing is almost always implemented by dividing the data between processors (data decomposition); the richness here is illustrated in Table I which lists the members of space that are decomposed for different use cases. Of course these sources of parallelism are broadly applicable outside the 51 use cases from which they were extracted. In Table II, we identify use case features for 15 use cases and map these to Ogre facets. The second column maps to the use case that illustrates this feature; note these are not exclusive so any one use case will illustrate many features.

For commonly used machine learning applications, there is an interesting distinction between what is termed Local (LML) or Global machine learning (GML) in Table II. In LML, there is parallelism over items of Table I and machine learning is applied separately to each item; needed machine learning parallelism is limited, typified by use of accelerators (GPU). In GML, the machine learning is applied over the full dataset with MapReduce, MPI or an equivalent. Typically GML comes from maximum likelihood or χ^2 with a sum over the data items - documents, sequences, items to be sold, images and often links (point-pairs). Usually GML is a sum of positive numbers, as in least squares, and is illustrated by algorithms like PageRank, clustering/community detection, mixture models, topic determination, Multidimensional scaling, and (Deep) Learning that constructs a learning network integrating all images.

3. TOWARDS A HIGH-PERFORMANCE BIG DATA SOFTWARE (HPBDS) ENVIRONMENT

As alluded to earlier, the HPC-ABDS [Jha et al. 2014b] approach was partially inspired by the NIST big data initiative [NIST 2013a] that generated a collection of 71 use cases as well as a taxonomy, reference architecture, roadmap and study of security and privacy. Later [Chang 2014] meetings identified use case patterns and mapped them to the NIST reference architecture. Figure 2 summarizes the ideas in an HPC-ABDS hourglass. To achieve high performance on data analysis, on the systems side, we have two principles: the Kaleidoscope of Apache Big Data Stack with 120 projects (see Fig. 5) has important broad functionality with a vital large support organization; HPC including MPI has striking success in delivering high performance,

Table II. Some Features of NIST Use Cases

Abbrev.	#	Description
PP	26	Pleasingly Parallel or Map Only
MR	18	Classic MapReduce MR (add MRStat below for full count)
MRStat	7	Simple version of MR where key computations are simple reduction as found in statistical averages such as histograms and averages
MRIter	23	Iterative MapReduce or MPI
Graph	9	Complex graph data structure needed in analysis
Fusion	11	Integrate diverse data to aid discovery/decision making; could involve sophisticated algorithms or could just be a portal
Streaming	41	Some data comes in incrementally and is processed this way
Classify	30	Classification: divide data into categories
S/Q	12	Index, Search and Query
CF	4	Collaborative Filtering for recommender engines
LML	36	Local Machine Learning (Independent for each parallel entity)
GML	23	Global Machine Learning: Deep Learning, Clustering, LDA, PLSI, MDS, Large Scale Optimizations as in Variational Bayes, MCMC, Lifted Belief Propagation, Stochastic Gradient Descent, L-BFGS, Levenberg-Marquardt. Can call EGO or Exascale Global Optimization with scalable parallel algorithm
	51	Workflow: Universal so no label
GIS	16	Geotagged data and often displayed in ESRI, Microsoft Virtual Earth, Google Earth, GeoServer etc.
HPC	5	Classic large-scale simulation of cosmos, materials, etc. generating (visualization) data
Agent	2	Simulations of models of data-defined macroscopic entities represented as agents

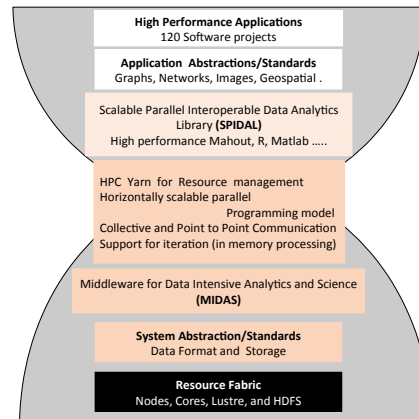


Fig. 2. HPC-ABDS

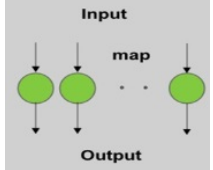
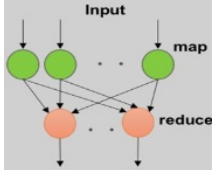
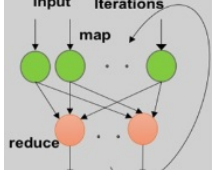
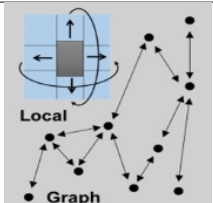
though with a fragile sustainability model. There are key systems abstractions where Apache approach needs careful integration with HPC in areas of resource management, storage, programming model (for horizontal scaling or parallelism), collective and point-to-point communication, support of iteration, and richness of data interface. In application areas, we define application abstractions to support Graphs/networks, Geospatial, Genes and Images.

Fox and Luckow [2014] identified a set of Ogres (big data patterns) covering big data analytics with multiple facets including specific algorithms, problem architecture and its features, application class and data source structure. Here we exploit these facets to identify the programming models, data source models and overall end-to-end application models that MIDAS needs to support.

Based upon an analysis of the Ogres, we identify the importance of 5 parallel programming models. They are supported by HPC-ABDS:

- PM1) Pleasingly Parallel (PP) includes many cases where there is sophisticated local machine learning applied in parallel – as in parallel image processing without global optimizations.
- PM2) Search (Srch) includes collaborative filtering (in Mahout), motif (meme) detection in graph (network) algorithms, and spatial relationship based queries for spatial data, and is implemented using classic MapReduce or non-iterative Giraph.
- PM3) Iterative MapReduce or Map-Collective using Collective Communication are seen in many global machine learning algorithms applied over the complete distributed dataset and are illustrated by clustering and dimensionality reduction using parallel linear algebra at their core.
- PM4) Iterative Giraph is Map-Communication with point-to-point communication and includes graph algorithms such as maximum clique, connected component, finding diameter, and community detection. The problems differ in the difficulty of determining the data partitioning and this classic parallel load balancing issue can need sophisticated runtime techniques.
- PM5) Asynchronous thread-based graph algorithms. These are illustrated by shortest path and betweenness centrality algorithms for shared memory machines and we do not integrate them into HPC-ABDS in this proposal.

Table III. Distinctive Software/Hardware Architectures for Data Analytics

1. Pleasingly Parallel (Map Only)		Includes local machine learning (LML) as in parallel decomposition over items and applying data processing to each item. Hadoop could be used as well as other High Throughput Computing or Many task tools
2. Classic Map-Reduce		Includes MRStat, search applications and those using collaborative filtering and motif finding implemented using classic MapReduce (Hadoop)
3. Iterative Map-Collective		Iterative MapReduce using Collective Communication as needed in clustering - Hadoop with Harp, Spark etc.
4. Iterative Map-Communication		Iterative MapReduce such as Giraph with point-to-point communication, includes most graph algorithms such as maximum clique, connected component, finding diameter, community detection). Varies in difficulty of finding partitioning (classic parallel load balancing)
5. Shared (Large) Memory		Thread-based (event driven) graph algorithms such as shortest path and Betweenness centrality. Large memory applications

In Table III, we present 5 distinct problem architectures that map into 5 distinct system architectures to cover the Ogres. The first four architectures of Table III which correspond to the four forms of MapReduce that we have presented previously [Ekanayake et al. 2010a]. Note this only describes some core features of the facets [Fox and Luckow 2014]. There are many other issues that need to be addressed including support of workflow. In particular the architecture for the rapidly evolving field of streaming (distributed) data needs more work.

Note that we separate Map-Collective [Barrett et al. 1994; der Wijngaart et al. 2012] and Map-(Point to Point) Communication following the Apache projects Hadoop, Spark and Giraph that focus on these cases. These programming models or run times differ in communication style, application abstraction (key-value versus graph) and possible scheduling/load-balancing. HPC with MPI suggests that one could integrate into a single environment. This approach is illustrated by the Harp plug-in [Qiu and Zhang 2014] to Hadoop which supports both models.

SPIDAL will capture these common characteristics and requirements by identifying key abstractions; it will utilize capabilities of the underlying middleware that will be exposed via well-designed and engineered libraries. The MIDAS middleware implements these with high performance in an ABDS context. MIDAS is based on abstractions in the following areas: a) Software defined System, b) Storage layer including a spatial access abstraction, c) Scheduling layer using advances in multi-level and application-level scheduling, d) Collective layer that permits Map Collective generalization of Iterative MapReduce, e) Parallelism or Programming model which generalizes the popular Giraph and MPI SPMD models.

4. SCALABLE PARALLEL INTEROPERABLE DATA-ANALYTICS LIBRARY (SPIDAL)

This section summarizes the core algorithms proposed initially for SPIDAL.

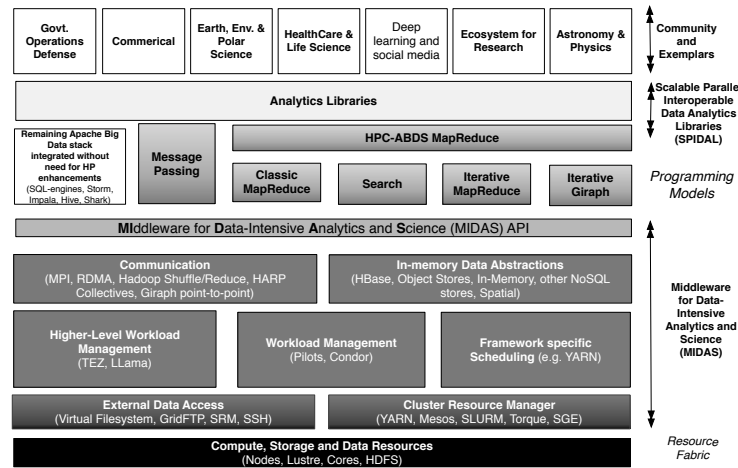


Fig. 3. SPIDAL: Library and Algorithms

4.1. Graph and Network Algorithms

4.1.1. Architectures for Graph Algorithms. Distributed-memory vs. shared-memory. For graph problems, researchers have developed both distributed-memory algorithms [N. Edmonds 2010; Alam et al. 2013; Arifuzzaman et al. 2013; Zhao et al. 2010, 2012] and shared-memory algorithms [Prountzos and Pingali 2013; Ediger et al. 2012; Bader and Cong 2005, 2004; Madduri et al. 2007, 2009]. In a distributed memory system, every processor has its own local memory, and data is partitioned so that each processor contains one partition in its memory. Since processors may need to communicate and exchange data with one another, poor locality is a major challenge for distributed-memory systems, causing communication overhead that can lead to decreased performance. A distributed memory system is good for graph problems with high locality. In a shared-memory system, data is stored in a common shared memory accessed by all processors and locality is not critical, although efficient thread parallelism may still be hard.

Message Passing Interface (MPI) vs. Giraph. MPI is a general-purpose distributed memory system for parallel programming, with efficient communication primitives. Efficient MPI implementations have been developed for a number of graph problems, which scale to very large networks, using problem-specific knowledge of the computation and communication patterns; this requires significant HPC expertise. In contrast, Giraph is easy to use, but does not allow easy access to partitioning and load balancing. Two main challenges in parallelization are: obtaining good estimates of the computation cost for each partition, and load balancing, both of which require problem specific insights [Nguyen et al. 2013]. This makes the problem of finding good partitions that minimize communication cost very challenging [Alam et al. 2013; Arifuzzaman et al. 2013].

4.1.2. Graph Algorithms in SPIDAL. To deal with the challenges posed by massive networks, we will develop new algorithmic techniques based on MapReduce and Giraph. In iterative MapReduce, we will explore techniques to decompose the problem and enable efficient information exchange through the reduce operation. The problem of finding network motifs/subgraphs is well-studied [Zhao et al. 2010, 2012; Alon et al. 2008; Aravind and Raman 2002; Gonen and Shavitt 2009; Milo et al. 2002; Qin and Gao 2012] with sequential algorithms, although [Ribeiro et al. 2012] addresses a shared-memory architecture. We developed distributed-memory parallel algorithms [Zhao et al. 2010, 2012]; however, these algorithms are for a few special classes of motifs, e.g., trees. We propose to develop Giraph-based distributed-memory parallel algorithms for a more general class of motifs. We will also build on the techniques of Karloff et al. [2010] to explore efficient MapReduce implementations for other PRAM algorithms with super-quadratic key-value space complexity, and identify problem classes which do not scale in MapReduce. For such problems, we will use Giraph. We will build on the general conversion theorem [Klauck et al. 2013] to develop implementations of algorithms for problems such as connectivity, subgraph enumeration and random graph generation in Giraph. Some of the results in [Klauck et al. 2013] yield constant factor or logarithmic approximation algorithms for problems such as shortest paths in near optimal time. We will explore the complexity of optimal algorithms for these problems.

Community detection problems (including clustering approaches) have only received serious attention recently. Current methods [Satuluri et al. 2011; Meyerhenke et al. 2011; Satuluri and Parthasarathy 2009; Fortunato 2010; Sarkar and Dong 2011; Zhang et al. 2009] do not scale well for large networks, including parallel algorithms [Meyerhenke et al. 2011; Satuluri and Parthasarathy 2009; Fortunato 2010; Sarkar and Dong 2011; Zhang et al. 2009] for shared-memory architecture that cannot support massive-scale networks. We propose to develop scalable and distributed-

memory parallel algorithms using Giraph. David Baders group (Georgia Tech) developed parallel algorithms for some other network problems, such as shortest path and betweenness centrality [Madduri et al. 2007; Bader and J. JJ 1996; Bader and Madduri 2008; Madduri and Bader 2009; Bader 2010; Green et al. 2012; Jiang et al. 2009]. Edmonds, Hoefler and Lumsdaine [N. Edmonds 2010] recently developed a space-efficient parallel algorithm for computing betweenness centrality in distributed-memory systems, while efficient sequential algorithms have been fashioned for selected network analytics problems such as diameter, page rank, and counting triangles, at CMU (Christos Faloutsos) [Faloutsos 2012; Kang et al. 2009, 2011; Tsourakakis et al. 2009] and Sandia National Lab [Zhang et al. 2009; Seshadhri et al. 2012a,b]. There also are sequential libraries for network analytics such as Pajek [Batagelj and Mrvar 1998], Pegasus [Faloutsos 2012] (Christos Faloutsos), SNAP [Leskovec 2012] (Jure Leskovec at Stanford), NetworkX [Hagberg et al. 2008] (at Los Alamos National Lab) and statnet [Handcock et al. 2003]. Similar libraries for parallel graph algorithms are needed to work with emerging massive networks.

4.2. Spatial Queries and Spatial Analytics Algorithms Related Work

Spatial Data Management Systems (SDBMS) have major limitations on managing and querying large-scale spatial data. SDBMSs [pro 2013f,i,c,h] rely on parallel DBMS architectures such as a shared nothing architecture [Mehta and DeWitt 1995; Patel et al. 1997; pro 2013j,d,g] to scale out. Parallel SDBMSs through partitioning are not optimized for computationally intensive operations such as geometric computations [Wang et al. 2012], and lack spatial partitioning to balance data and task loads [pro 2013b]. Data loading overhead is another major bottleneck [Pavlo et al. 2009; Aji et al. 2013; Wang et al. 2013]. GIS systems [Chang 2003; pro 2013a] often use SDBMS as the backend spatial engine. Work in [Cary et al. 2009; Zhang et al. a,b; Liu et al. 2010; Pozdnoukhov and Kaiser 2011; Li et al. 2012; Guo et al. 2010; Ballesteros et al. 2011; Ma et al. 2009; Akdogan et al. 2010] tries to tackle specific spatial algorithms using MapReduce, and ongoing MapReduce spatial querying systems including [Eldawy and Mokbel 2013; Daszak 2000]. Our Hadoop-GIS [Aji et al. 2013; pro 2013e] provides a general framework for spatial queries and analytics with MapReduce. Hadoop-GIS is integrated into Apache Hive to support declarative spatial queries.

Spatial data has multiple dimensions and there is no concept of a key in a spatial space for data partitioning or task partitioning. We will partition space into tiles, and break down time-consuming spatial operations into smaller tasks on them, running in parallel in HPC-ABDS MapReduce. Thus a tile serves as a key, and the objects of a tile are the value. To avoid data skew, density-aware spatial partitioning methods are provided. Spatial query processing then becomes a problem of designing query methods that can run on tiles independently while preserving correct semantics. A generic framework for distributed spatial data processing has the following steps. Initialization includes data partitioning, uploading and indexing. For each spatial query, the index is used to identify regions of interest for the query. After that, spatial query processing is executed independently for each tile, in parallel. During each tile based query, on-demand indexing can be provided to accelerate queries. Then a normalization step will be performed to fix results due to partitioning. Most spatial processing methods can be mapped into the above patterns, and spatial clustering follows global machine learning patterns including nearest neighbor search, density and statistics-based spatial clustering and regression analysis.

4.3. Core Image Processing Algorithms Related Work on Pathology informatics

The availability of advanced scanners has fueled efforts in whole slide image analysis. Algorithms have been developed for detection and extraction of features from whole

slide images. Data models and databases for representation and management of microscopy image data have also been developed [Goldberg et al. 2005; Schiffmann et al. 2006; Martone et al. 2008, 2003; Hayden et al. 2007; Wang et al. 2010]. ITK and VTK are two well-known libraries mainly designed to support Radiology images but sequential processing can take hours for an image. Several recent research efforts, have investigated the use of GPUs, distributed memory parallel machines, and Grid computing to analyze large images and datasets. SPIDAL will build on and complement this and will enable much larger scale investigations. The abstractions will enable interoperability of image analysis with other methods such as clustering, motif detection, complexity reduction, and network abstraction, leading to broader deployment and novel analyses.

While computer vision has a 50 year history, only recently have researchers considered large-scale datasets driven by the dramatic rise of online social media. For instance, papers in 3D reconstruction [Crandall et al. 2011; Agarwal et al. 2009; Crandall et al. 2011; Crandall and Snavely 2012], visual geolocation [Li et al. 2009; Hays and Efros 2008], denoising [Hays and Efros 2007], scene classification [Xiao, Hays, Ehinger, Oliva, and Torralba Xiao et al.], and object recognition [University 2014] use millions of images from Flickr. While there are established libraries for single machine computer vision, like OpenCV, Matlab, CImg, VLFeat, and ImageJ, there are no widely adopted libraries for large-scale computer vision, leading researchers to roll out their own ad-hoc libraries (although some nascent parallel libraries are under development, like CloudCV (Virginia Tech) and Hadoop Image Processing Interface (Virginia)).

In SPIDAL, we will provide a generic high-performance framework that frees vision researchers from considering system-level issues. Our core image processing library encapsulates common operations employed by computer vision, pathology informatics, spatial informatics, and radar informatics. These routines include low-level image preprocessing like convolution, edge detection, color correction; core primitive object detection (e.g. Hough transform) and segmentation (MinCuts); low-level feature extracting [Lowe 2004; Dalal and Triggs 2005; Torralba, Murphy, Freeman, and Rubin Torralba et al.]; registering images with 3D models; object matching; and 3D feature extraction.

The library also includes derived routines for specific domains. For example, in pathology and spatial informatics, whole slide images are extremely large (10 billion pixels per image) and cannot fit in memory requiring tiling as described above. This is not the case in computer vision, where images are typically from consumer cameras and the challenge is number of images and not their individual size. We will provide a generic parallel image processing pipeline that supports different image analysis algorithms with HPC-ABDS. For large images, we provide tile-based partitioning with buffered tiles at boundaries. Tiled images can be processed independently in parallel for image pre-processing, segmentation, and feature extraction. The results from tiles are converted into vector-based geometric shapes and structural features with correction for boundary object normalization. For 3D images, one performs 3D object grouping from 2D segmentation results, and 3D feature extraction, which again can be run in parallel.

4.4. Existing General Libraries and Machine Learning

Mahout [Apache Mahout 2012] is an open source project that builds machine learning libraries based on Hadoop MapReduce. It supports supervised and unsupervised learning algorithms such as Collaborative Filtering, K-means Clustering, Latent Dirichlet Allocation, and Naïve Bayes Classifiers. While it offers fast implementations for some algorithms like Naïve Bayes, for problems with iteration (like clustering) it sacrifices performance for scalability by writing intermediate data out to disk after each

iteration. For several algorithms, like Hidden Markov Models, Multilayer Perceptron, and Logistic Regression, it does not provide any parallel implementation at all (just a single-machine version).

We have studied Mahout [Apache Mahout 2012] carefully, and have chosen key algorithms to include in SPIDAL, including SVD, Random Forests, SVMs, K-means, HMM, and LDA. For some of these, either IU or Mahout already has parallelized code, which we will simply port to HPC-ABDS to make them available in our framework. Other Mahout algorithms are either not parallelized or require iteration (which in Hadoop requires costly disk I/O); for these, HPC-ABDS will offer much higher-performance implementations.

The Spark MLlib (MLbase) library [MLlib 2014] is another promising effort, offering a few algorithms (SVM, K-means, gradient descent) supported by a small community. We will monitor developments in that library as it matures and port from there if possible. We will also port clustering routines from R [pro 2012] to test our ability to move algorithms from this important library to HPC-ABDS.

Parallelized implementations from IU include clustering with a sophisticated deterministic annealing (DA) method which was originally developed with Rose [Rose 1998; Rose et al. 1990] but with recent parallel implementations by Fox [2013]; Fox et al. [2009a]. These codes cover the cases of both metric and non-metric spaces and have been used extensively in bioinformatics [Qiu et al. 2011; Fox et al. 2009b; Ruan et al. 2012a; Stanberry et al. 2012; Ruan et al. 2012b; Qiu et al. 2010; Hughes et al. 2012; Ruan and Fox 2013; Ruan et al. 2014; Fruhwirth et al. 2011]. These will build upon [Choi 2012], wherein was developed DA improvements to mixture model approaches to find hidden factors and we will also implement this in HPC-ABDS. This was successfully applied to find the context of HPC jobs and improve pre-fetching [Choi et al. 2012].

The non-metric case uses Multi-Dimensional Scaling (MDS) using either SMACOF improved with DA [Ruan and Fox 2013; Bae 2012], or a method using nonlinear χ^2 minimization [Kearsley et al. 1995] which is typically slower but allows a more flexible weighted objective function. There are several methods for dimension reduction with metric spaces and we will deploy in SPIDAL, e.g., a parallel DA enhanced GTM (Generative Topographic Mapping) from [Choi 2012; Choi et al. 2011].

5. MIDDLEWARE FOR DATA-INTENSIVE ANALYTICS AND SCIENCE (MIDAS)

The aim of MIDAS is to provide a scalable runtime system and appropriate resource management abstractions enabling SPIDAL, and thereby Big Data applications. As identified by the Ogres and their facets, Big Data applications have a wide range of characteristics requiring different programming models and different forms of parallelism, from the execution of large numbers of loosely coupled tasks, to in-memory caching for iterative MapReduce, to parallel linear algebra computations to support GML algorithms such as Deep Learning. MIDAS provides the underlying resource management middleware and heterogeneous infrastructure access layer which will support SPIDAL libraries to work efficiently across these application types over a range of resources.

Figure 4 shows the architecture of MIDAS which has two primary design objectives: (i) Provide high-level abstractions (e.g. scalable data processing, inter-process communication and storage supporting both query and analysis), so as to hide details of different lower level implementations (e.g. for accessing data or resources via HPC schedulers such as SLURM, Big Data schedulers, such as YARN [Hortonworks 2014] or Cloud backends like Amazon and Google), (ii) provide a flexible middleware to support four key programming models identified in Table III: (PM1) Pleasingly parallel, (PM2) Search using Classic MapReduce, (PM3) Iterative MapReduce with Collectives

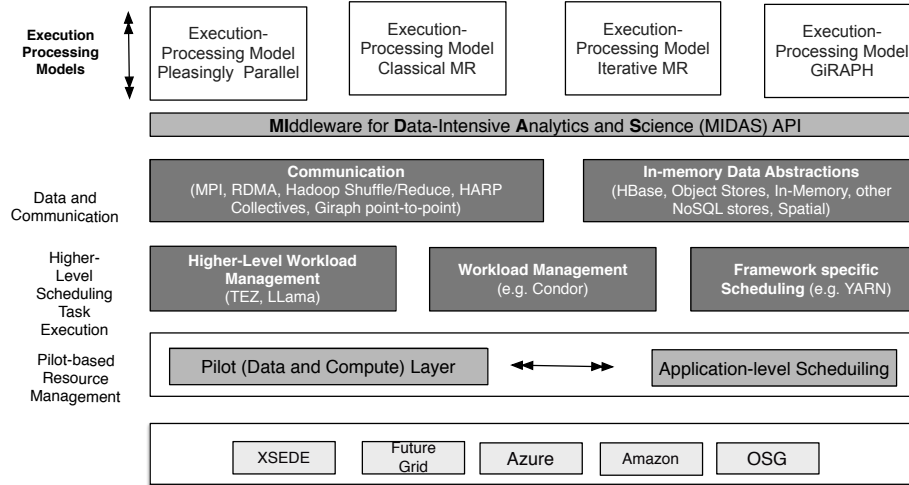


Fig. 4. MIDAS Layered Architecture View: The Pilot-Layer provides the basis for higher-level MIDAS abstractions supporting e.g. access to heterogeneous compute and data resources and in-memory caching for the iterative MapReduce programming model.

and (PM4) Iterative Graph Processing. These programming models require different runtime systems to extract performance on different platforms varying from clouds to HPC. MIDAS will provide the scalable runtime system to support these programming models via appropriate execution-processing capabilities on different platforms.

To support these programming models, we propose several abstractions: an inter-process communication layer and a layer handling computation, for example offering different data abstractions for Hadoop (key-value, dataflow) and Giraph (graph). Communication abstractions enable the coordination and exchange of data between tasks. In particular iterative MapReduce tasks need collective operations while graph processing largely needs point-to-point communication. We have already shown that using classic MPI techniques can provide a collective layer that outperforms existing (iterative) MapReduce approaches on both cloud and HPC environments [Jha et al. 2014b; Qiu and Zhang 2013, 2014]. We will expand this concept in MIDAS covering all 4 programming models and defining the HPC-ABDS MapReduce model shown in system architecture. The communication layer is designed in a pluggable, infrastructure agnostic way and can be used within Hadoop and HPC applications, e.g. based on the Pilot abstraction.

In-Memory abstractions can be used to implicitly facilitate the data exchange between multiple tasks, or to efficiently retain data between generations of tasks. This is essential to efficiently support iterative processing or graph processing (patterns (MP3) and (MP4)). In particular, Harp [Qiu and Zhang 2014] building on Iterative MapReduce Twister [Ekanayake et al. 2010b] is an open source collective communication library which can be plugged into Hadoop. With this plug-in, Map-Reduce jobs can be transformed into Map-Collective jobs and users can invoke efficient in-memory message passing operations such as collective communication (e.g. broadcast and group-by) and point-to-point communication directly in Map tasks. For the first time, Map-Collective brings high performance to ABDS in a clear communication abstraction, which did not exist before in the Hadoop ecosystem. We expect Harp to equal MPI performance with straightforward optimizations. In addition, Harp improves the expressiveness in big data processing with the support of data abstraction types such

as arrays, key-values, and graphs, with related collective communication operations on top of each type. Several applications have been developed based on Harp framework, including K-means clustering, Multi-dimensional scaling and Page Rank. Harp being based on Hadoop means it leverages better sustainability and fault tolerance properties. The above is illustrative of a critical design challenge that faces MIDAS: balancing performance with flexibility.

In earlier Hadoop versions, it was necessary to retrofit non-MapReduce applications (as in for pleasingly parallel processing and machine learning) into the rigid MapReduce programming model. This is not necessary anymore for Hadoop-2 (YARN) [Hortonworks 2014], as YARN can co-locate and run any application (MPI, MapReduce, iterative MR, graph, or based on any other library/framework) on a Hadoop cluster. Ensuring support for YARN on HPC platforms such as XSEDE will thus be an important objective. However, while YARN removes the programming model limitation, various challenges remain: (i) the current usability of YARN on HPC environments is limited (as it was designed as a system-level framework), (ii) resource containers cannot easily be re-used across multiple tasks, and (iii) interoperability with HPC environments (Blue Waters, to a variety of XSEDE resources). To address these limitations, we propose to build upon and extend the Pilot-abstraction, which is a proven approach in supporting data-parallel tasks on top of heterogeneous infrastructures [Luckow et al. 2012, 2014]. Using Pilot-Jobs, we allow processing frameworks to interoperably exploit YARN, common HPC and cloud resource managers. Further, we will extend the Pilot-Data abstraction to support in-memory processing required for iterative MapReduce in an infrastructure and data source agnostic way.

Looking at data source aspects of the Ogres (big data patterns), though most applications use a collection of files we expect a growing interest in object stores and will support both with the *Pilot-Data* abstraction. Several search problems need customizable dynamic indices. Also relevant is in-memory data storage, data format, data partitioning and the access model. Thus, we will extend a canonical abstraction of Pilot-Jobs to Pilot-Data to support interoperable access of data stored in Hadoop, databases, and NOSQL as well as other types of HPC storage (Lustre, iRODS, etc.) as needed.

In addition to data access, MIDAS will support the common data processing patterns (partition, process, merge, etc.) on top of datasets managed by Pilot-Data. As demonstrated by the evolution to YARN, there is an increasing need to provide application specific scheduling and resource management control; the Pilot abstraction has demonstrated such capabilities. Application-level scheduling as provided by the Pilot-layer will be an essential tool to integrate library resource usage modes (see processing patterns) with resource allocation/usage. This will be required for the network science community, where flexible integration with a logical resource partitioner is important. For example, we will have to examine the trade-offs of partitioning at the MPI level or at the Giraph programming model depending upon data size and architecture. In general, MIDAS is intended to be agnostic to workflow models and will support gateways, scripting and specific workflow tools. Auto-tuning (akin to ATLAS for linear algebra software) and load balancing for each execution processing model to optimize data locality and communication are critical capabilities to provide on analytical applications. Last but not least, the infrastructure access layer will use the existing capabilities of SAGA, which provides most resource access functionality using a standards-based layer. This will ensure that the new software footprint of MIDAS will be small, whilst providing reuse of established and existing building blocks.

6. DISCUSSION AND CONCLUSION

As alluded to in the title, HPC-ABDS is the first attempt led by the Indiana-Rutgers collaboration at creating a software system to meet Big data analytics, i.e., a high-

performance Big Data Stack (HPBDS). There are several incipient and emerging efforts at addressing the requirements for big data on high-end systems. HPC-ABDS is distinguished by its efforts to address the requirements of a well defined set of application patterns and their primary characteristics (referred to as Ogres), that are commonly required of data-analytics applications, via the construction of libraries that work across a range of high-end computing platforms, including clouds. We believe the design of such SPIDAL libraries – based upon an analysis of Big Data Ogres, as opposed to a limited set of applications – along with a careful co-design of the resource management MIDAS layer makes HPC-ABDS distinctive.

In addition to the NIST use cases, HPC-ABDS will enable multiple communities to use a core set of libraries (SPIDAL – graphs, imaging, spatial, remote sensing, HPC simulation and modeling) on top of an abstractions-based (Job, data, communication, in-memory operations, parallelism) high performance middleware MIDAS. By integrating the two fundamental building blocks – MIDAS and SPIDAL, HPC-ABDS will provide new levels of scalability and application performance along multiple dimensions by combining application expertise, the broad Apache Big Data stack and best practice HPC. Specifically, HPC-ABDS will: (i) implement resource management capabilities of MIDAS using job and data abstractions, as well as scalable and reusable fine-grained building blocks for HPC/high-end resources, (ii) define library interfaces (SPIDAL, with appropriate language bindings) to these fine-grained building blocks, (iii) provide scalable and parallel analytic libraries by integrating the aforementioned interfaces, the fine-grained building blocks with the resource management middleware capabilities of MIDAS.

SPIDAL libraries will be provided with a set of simple benchmark kernels. The initial focus will be on data analytics libraries and needed abstractions built on a skeletal MIDAS middleware whose key features have already been demonstrated.

Kaleidoscope of Apache Big Data Stack

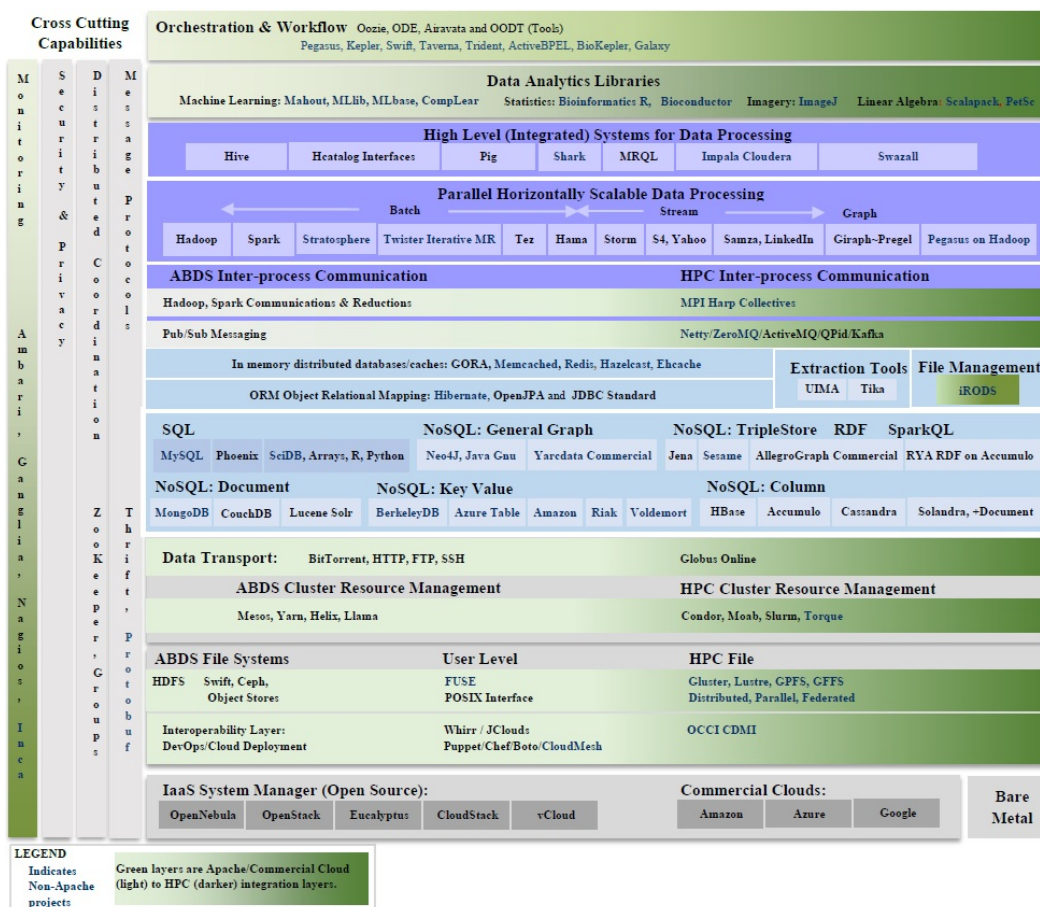


Fig. 5. For updated figure see: <http://hpc-abds.org/kaleidoscope/>

7. APPENDIX

REFERENCES

2012. CRAN Task View: Cluster Analysis and Finite Mixture Models. (July 21 2012). <http://cran.cnr.berkeley.edu/web/views/Cluster.html>
- 2013a. ArcGIS. (2013). <http://www.esri.com/software/arcgis>
- 2013b. Database Partitioning, Table Partitioning, and MDC for DB2 9. (2013). <http://www.redbooks.ibm.com/redbooks/pdfs/sg247467.pdf>
- 2013c. Delivering Location Intelligence with Spatial Data. (2013). <http://www.microsoft.com/sql/techinfo/whitepapers/spatialdata.msp>
- 2013d. Greenplum. (2013). <http://www.greenplum.com/products/greenplum-database>
- 2013e. Hadoop-GIS Wiki. (2013). <https://web.cci.emory.edu/confluence/display/HadoopGIS>
- 2013f. IBM DB2 Spatial. (2013). <http://www-01.ibm.com/software/data/spatial/>
- 2013g. IBM Netezza. (2013). <http://www-01.ibm.com/software/data/netezza/>

- 2013h. Oracle Spatial and Oracle Locator. (2013). <http://www.oracle.com/us/products/database/options/spatial/overview/index.html>
- 2013i. PostGIS. (2013). <http://postgis.refractory.net/>
- 2013j. Teradata. (2013). <http://www.teradata.com/>
- Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. 2009. Building Rome in a day. (Sept. 29-Oct. 2 2009). DOI: <http://dx.doi.org/10.1109/ICCV.2009.5459148>
- Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel Saltz. 2013. Hadoop-GIS: A High Performance Spatial Data Warehousing System Over MapReduce. *Proc. VLDB Endow.* 6, 11 (2013), 1009–1020.
- Afsin Akdogan, Ugur Demiryurek, Farnoush Banaei-Kashani, and Cyrus Shahabi. 2010. Voronoi-Based Geospatial Query Processing with MapReduce. (2010).
- Maksudul Alam, Maleq Khan, and Madhav V. Marathe. 2013. Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model. (2013). DOI: <http://dx.doi.org/10.1145/2503210.2503291>
- N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. Sahinalp. 2008. Biomolecular network motif counting and discovery by color coding. *Bioinformatics* 24, 13 (2008), i241.
- Apache Mahout. 2012. Apache Mahout Scalable machine learning and data mining. <http://mahout.apache.org/>. (2012).
- V. Aravind and V. Raman. 2002. Approximate counting of small subgraphs of bounded treewidth and related problems. (2002).
- Shaikh Arifuzzaman, Maleq Khan, and Madhav Marathe. 2013. PATRIC: a parallel algorithm for counting triangles in massive networks. (2013). DOI: <http://dx.doi.org/10.1145/2505515.2505545>
- D.A. Bader. 2010. Analyzing Massive Social Networks using Multicore and Multi-threaded Architectures. *Facing the Multicore-Challenge: Aspects of New Paradigms and Technologies in Parallel Computing, Lecture Notes in Computer Science* 6310, 1 (2010).
- D.A. Bader and G. Cong. 2004. Fast Shared-Memory Algorithms for Computing the Minimum Spanning Forest of Sparse Graphs. (April 26-30 2004).
- D.A. Bader and K. Madduri. 2008. A Graph-Theoretic Analysis of the Human Protein-Interaction Network Using Multi-core Parallel Algorithms. *Parallel Comput.* 34, 11 (2008), 627–639.
- David A. Bader and Guojing Cong. 2005. A fast, parallel spanning tree algorithm for symmetric multiprocessors (SMPs). *J. Parallel Distrib. Comput.* 65, 9 (2005), 994–1006. DOI: <http://dx.doi.org/10.1016/j.jpdc.2005.03.011>
- D. A. Bader and . J. JJ. 1996. Parallel Algorithms for Image Histogramming and Connected Components with an Experimental Study. *J. Parallel and Distrib. Comput.* 35, 2 (1996), 173–190.
- Seung-Hee Bae. 2012. *SCALABLE HIGH PERFORMANCE MULTIDIMENSIONAL SCALING*. Thesis. <http://grids.ucsf.edu/ptliupages/publications/SeungheeBae.Dissertation.pdf>
- Jaime Ballesteros, Ariel Cary, and Naphtali Rishe. 2011. SpSJoin: Parallel Spatial Similarity Joins. (2011). DOI: <http://dx.doi.org/10.1145/2093973.2094054>
- R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA.
- V. Batagelj and A. Mrvar. 1998. Pajek - Program for Large Network Analysis. *Connections* 21, 2 (1998), 47–57. <http://vlado.fmf.uni-lj.si/pub/networks/doc/pajek.pdf>

- Ariel Cary, Zhengguo Sun, Vagelis Hristidis, and Naphtali Rishe. 2009. Experiences on Processing Spatial Data with MapReduce. (2009). DOI: http://dx.doi.org/10.1007/978-3-642-02279-1_24
- Kang-Tsung Chang. 2003. *Introduction to Geographic Information Systems*. Prentice Hall PTR. 384 pages.
- Wo Chang. 2014. ISO/IEC JTC 1 Study Group on Big Data. http://jtc1bigdatasg.nist.gov/flyer_BigDataEcosystem_Workshop_US.pdf. In *1st Big Data Interoperability Framework Workshop: Building Robust Big Data Ecosystem*, Vol. 2014. NIST.
- Jong Youl Choi. 2012. *Unsupervised Learning Of Finite Mixture Models With Deterministic Annealing For Large-scale Data Analysis*. Thesis. <http://grids.ucs.indiana.edu/ptliupages/publications/damix.final.v1.pdf>
- Jong Youl Choi, Mohammad H. Abbasi, David Pugmire, Scott Klasky, Judy Qiu, and Geoffrey Fox. 2012. Mining Hidden Mixture Context With ADIOS-P To Improve Predictive Pre-fetcher Accuracy. (October 8-12 2012). [http://grids.ucs.indiana.edu/ptliupages/publications/hcming\(1\).pdf](http://grids.ucs.indiana.edu/ptliupages/publications/hcming(1).pdf)
- Jong Youl Choi, Seung-Hee Bae, Judy Qiu, Bin Chen, and David Wild. 2011. Browsing Large Scale Cheminformatics Data with Dimension Reduction. *Concurr. Comput. : Pract. Exper.* Special Issue on ECMLS2010 (2011). <http://grids.ucs.indiana.edu/ptliupages/publications/plotviz.v6.pdf>
- David Crandall, Andrew Owens, Noah Snavely, and Daniel P. Huttenlocher. 2011. Discrete-continuous optimization for large-scale structure from motion. (2011).
- David Crandall and Noah Snavely. 2012. Modeling people and places with internet photo collections. *Commun. ACM* 55, 6 (2012), 52–60. DOI: <http://dx.doi.org/10.1145/2184319.2184336>
- Navneet Dalal and Bill Triggs. 2005. Histograms of Oriented Gradients for Human Detection. (2005). DOI: <http://dx.doi.org/10.1109/cvpr.2005.177>
- P. Daszak. 2000. Emerging Infectious Diseases of Wildlife— Threats to Biodiversity and Human Health. *Science* 287, 5452 (2000), 443–449. DOI: <http://dx.doi.org/10.1126/science.287.5452.443>
- Rob F. Van der Wijngaart, Srinivas Sridharan, and Victor W. Lee. 2012. Extending the BT NAS parallel benchmark to exascale computing. (2012).
- D. Ediger, K. Jiang, E.J. Riedy, and D.A. Bader. 2012. GraphCT: Multithreaded Algorithms for Massive Graph Analysis. *IEEE Transactions on Parallel & Distributed Systems* (2012).
- Jaliya Ekanayake, Thilina Gunarathne, Judy Qiu, Geoffrey Fox, Scott Beason, Jong Youl Choi, Yang Ruan, Seung-Hee Bae, and Hui Li. 2010a. *Applicability of DryadLINQ to Scientific Applications*. Report. Community Grids Laboratory, Indiana University.
- J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. Bae, J. Qiu, and G. Fox. 2010b. Twister: A Runtime for iterative MapReduce. (2010). <http://grids.ucs.indiana.edu/ptliupages/publications/hpdc-camera-ready-submission.pdf>
- Ahmed Eldawy and Mohamed Mokbel. 2013. A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data. (2013).
- Christos Faloutsos. 2012. Project Pegasus Peta-Scale graph mining. (2012). <http://www.cs.cmu.edu/~pegasus/>
- S. Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75–174.
- Geoffrey Fox. 2013. Robust Scalable Visualized Clustering in Vector and non Vector Semimetric Spaces. *Parallel Processing Letters* 23, 2 (2013). DOI: <http://dx.doi.org/doi/abs/10.1142/S0129626413400069>
- Geoffrey Fox, Seung-Hee Bae, Jaliya Ekanayake, Xiaohong Qiu, and Huapeng Yuan. 2009a. *Parallel Data Mining from Multicore to Cloudy Grids*.

- IOS Press, Amsterdam. <http://grids.ucs.indiana.edu/ptliupages/publications/CetraroWriteupJune11-09.pdf>
- Geoffrey Fox, Xiaohong Qiu, Scott Beason, Jong Youl Choi, Mina Rho, Haixu Tang, Neil Devadasan, and Gilbert Liu. 2009b. Biomedical Case Studies in Data Intensive Computing. (December 1-4 2009). <http://grids.ucs.indiana.edu/ptliupages/publications/SALSACloudCompaperOct10-09.pdf>
- Judy Fox, Shantenu Jha and Andre Luckow. 2014. Towards an Understanding of Facets and Exemplars of Big Data Applications. (October 13-14 2014).
- Rudolf Fruhwirth, D. R. Mani, and Saumyadipta Pyne. 2011. Clustering with position-specific constraints on variance: Applying redescending M-estimators to label-free LC-MS data analysis. *BMC Bioinformatics* 12, 1 (2011), 358. <http://www.biomedcentral.com/1471-2105/12/358>
- I. G. Goldberg, C. Allan, J. M. Burel, D. Creager, A. Falconi, H. Hochheiser, J. Johnston, J. Mellen, P. K. Sorger, and J. R. Swedlow. 2005. The Open Microscopy Environment (OME) Data Model and XML file: open tools for informatics and quantitative analysis in biological imaging. *Genome Biol* 6, 5 (2005), R47. DOI:<http://dx.doi.org/gb-2005-6-5-r47> [pii];10.1186/gb-2005-6-5-r47
- M. Gonen and Y. Shavitt. 2009. Approximating the number of network motifs. (2009).
- O. Green, R. McColl, and D.A. Bader. 2012. A Fast Algorithm for Incremental Betweenness Centrality. (September 3-5 2012).
- Danhui Guo, Kaichao Wu, Jianhui Li, and Yuwei Wang. 2010. Spatial scene similarity assessment on Hadoop. (2010). DOI:<http://dx.doi.org/10.1145/1869692.1869700>
- A. A. Hagberg, D. A. Schult, and P. J. Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. (2008). http://conference.scipy.org/proceedings/scipy2008/paper_2/
- M. Handcock, D. Hunter, C. Butts, S. Goodreau, and M. Morris. 2003. statnet: Software tools for the Statistical Modeling of Network Data, Version 2.0. (2003). <http://statnetproject.org>
- Linda Hayden, Geoffrey Fox, and Prasad Gogineni. 2007. CYBERINFRASTRUCTURE FOR REMOTE SENSING OF ICE SHEETS. (June 4-8 2007). http://grids.ucs.indiana.edu/ptliupages/publications/TeraGrid07_paper.pdf
- James Hays and Alexei A. Efros. 2007. Scene completion using millions of photographs. *ACM Trans. Graph.* 26, 3 (2007), 4. DOI:<http://dx.doi.org/10.1145/1276377.1276382>
- James Hays and Alexei A. Efros. 2008. IM2GPS: Estimating Geographic Information from a Single Image. (2008).
- Hortonworks. 2014. Apache Hadoop YARN is a sub-project of Hadoop introduced in Hadoop 2.0. (2014). <http://hortonworks.com/hadoop/yarn/>
- Adam Hughes, Yang Ruan, Saliya Ekanayake, Seung-Hee Bae, Qunfeng Dong, Mina Rho, Judy Qiu, and Geoffrey Fox. 2012. Interpolative Multidimensional Scaling Techniques for the Identification of Clusters in Very Large Sequence Sets. *BMC Bioinformatics* 13(Suppl 2):S9, Special Issue of for Proceedings of GLBIO Great Lakes Bioinformatics Conference Ohio University Athens Ohio May 2-4 2011 (2012). DOI:<http://dx.doi.org/10.1186/1471-2105-13-S2-S9>
- S. Jha, M. Cole, D. Katz, O. Rana, M. Parashar, and J. Weissman. 2013. Distributed Computing Practice for Large-Scale Science & Engineering Applications. *Concurrency and Computation: Practice and Experience* 25, 11 (2013), 1559–1585. DOI:<http://dx.doi.org/10.1002/cpe.2897>
- Shantenu Jha, Neil Chue Hong, Simon Dobson, Daniel S. Katz, Andre Luckow, Omer Rana, and Yogesh Simmhan. 2014a. Introducing Distributed Dynamic Data-intensive (D3) Science: Understanding Applications and Infrastructure. (2014).

- Shantenu Jha, Judy Qiu, Andre Luckow, Pradeep Mantha, and Geoffrey C. Fox. 2014b. A Tale of Two Data-Intensive Approaches: Applications, Architectures and Infrastructure. <http://arxiv.org/abs/1403.1528>. (June 27- July 2 2014).
- K. Jiang, D. Ediger, and D.A. Bader. 2009. Generalizing k-Betweenness Centrality Using Short Paths and a Parallel Multithreaded Implementation. (September 22-25 2009).
- U Kang, Charalampos Tsourakakis, and Christos Faloutsos. 2009. PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations. (December 2009).
- U Kang, Charalampos E. Tsourakakis, Ana Paula Appel, Christos Faloutsos, and Jure Leskovec. 2011. HADI: Mining Radii of Large Graphs. *ACM Trans. Knowl. Discov. Data* 5, 2 (2011), 1–24. DOI: <http://dx.doi.org/10.1145/1921632.1921634>
- Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. 2010. A model of computation for MapReduce. (2010).
- Anthony J. Kearsley, Richard A. Tapia, and Michael W. Trosset. 1995. *The Solution of the Metric STRESS and SSTRESS Problems in Multidimensional Scaling Using Newtons Method*. Report. Rice University.
- Hartmut Klauck, Danupon Nanongkai, Gopal Pandurangan, and Peter Robinson. 2013. *The Distributed Complexity of Large-scale Graph Processing*. Report. <http://arxiv.org/abs/1311.6209>
- J. Leskovec. 2012. Stanford Network Analysis Project. (2012). <http://snap.stanford.edu/>
- Wengen Li, Weili Wang, and Ting Jin. 2012. *Evaluating Spatial Keyword Queries under the MapReduce Framework Database Systems for Advanced Applications*. Lecture Notes in Computer Science, Vol. 7240. Springer Berlin / Heidelberg, 251–261. DOI: http://dx.doi.org/10.1007/978-3-642-29023-7_26
- Yunpeng Li, David Crandall, and Daniel P. Huttenlocher. 2009. Landmark Classification in Large-scale Image Collections. (2009).
- Yan Liu, Kaichao Wu, Shaowen Wang, Yanli Zhao, and Qian Huang. 2010. A MapReduce Approach to Gi*(d) Spatial Statistic. (2010). DOI: <http://dx.doi.org/10.1145/1869692.1869695>
- David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision* 60, 2 (2004), 91–110. DOI: <http://dx.doi.org/10.1023/b:visi.0000029664.99615.94>
- Andre Luckow, Mark Santcroos, and Shantenu Jha. 2014. Pilot-Data: An Abstraction for Distributed Data. *J. Parallel and Distrib. Comput.* In press (2014). <http://arXiv.org/abs/1301.6228>
- Andre Luckow, Mark Santcroos, Ole Weidner, Andre Merzky, Pradeep Mantha, and Shantenu Jha. 2012. P*: A Model of Pilot-Abstractions. (2012). DOI: <http://dx.doi.org/10.1109/eScience.2012.6404423>
- Qiang Ma, Bin Yang, Weining Qian, and Aoying Zhou. 2009. Query Processing of Massive Trajectory Data Based on Mapreduce. (2009). DOI: <http://dx.doi.org/10.1145/1651263.1651266>
- K. Madduri and D.A. Bader. 2009. Compact Graph Representations and Parallel Connectivity Algorithms for Massive Dynamic Network Analysis. (May 25-29 2009).
- K. Madduri, D.A. Bader, J.W. Berry, and J.R. Crobak. 2007. An Experimental Study of A Parallel Shortest Path Algorithm for Solving Large-Scale Graph Instances. (January 6 2007).
- Kamesh Madduri, David Ediger, Karl Jiang, David A. Bader, and Daniel Chavarria-Miranda. 2009. A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. (2009). DOI: <http://dx.doi.org/10.1109/ipdps.2009.5161100>

- M. E. Martone, J. Tran, W. W. Wong, J. Sargis, L. Fong, S. Larson, S. P. Lamont, A. Gupta, and M. H. Ellisman. 2008. The cell centered database project: an update on building community resources for managing and sharing 3D imaging data. *J Struct Biol* 161, 3 (2008), 220–31. DOI: [http://dx.doi.org/S1047-8477\(07\)00254-7\[pil\];10.1016/j.jsb.2007.10.003](http://dx.doi.org/S1047-8477(07)00254-7[pil];10.1016/j.jsb.2007.10.003)
- M. E. Martone, S. Zhang, A. Gupta, X. Qian, H. He, D. L. Price, M. Wong, S. Santini, and M. H. Ellisman. 2003. The cell-centered database: a database for multiscale structural and protein localization data from light and electron microscopy. 1, 4 (2003), 379–95. DOI: [http://dx.doi.org/NI:1:4:379\[pil\];10.1385/NI:1:4:379](http://dx.doi.org/NI:1:4:379[pil];10.1385/NI:1:4:379)
- Manish Mehta and David J. DeWitt. 1995. Managing Intra-operator Parallelism in Parallel Database Systems. (1995).
- Henning Meyerhenke, David Ediger, and David A. Bader. 2011. Parallel Community Detection for Massive Graphs. (September 2011).
- R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824.
- MLlib. 2014. Machine Learning Library (MLlib). <http://spark.apache.org/docs/0.9.0/mllib-guide.html>. (2014).
- A. Lumsdaine N. Edmonds, T. Hoefler. 2010. A space-efficient parallel algorithm for computing betweenness centrality in distributed memory. (2010).
- Donald Nguyen, Andrew Lenharth, and Keshav Pingali. 2013. A lightweight infrastructure for graph analytics. (2013). DOI: <http://dx.doi.org/10.1145/2517349.2522739>
- NIST. 2013a. Big Data Initiative Reports from V1. http://bigdatawg.nist.gov/V1-output_docs.php. (2013).
- NIST. 2013b. NIST Big Data Public Working Group (NBD-PWG) Use Cases and Requirements. <http://bigdatawg.nist.gov/usecases.php>, (2013).
- Jignesh Patel, JieBing Yu, Navin Kabra, Kristin Tufte, Biswadeep Nag, Josef Burger, Nancy Hall, Karthikeyan Ramasamy, Roger Lueder, Curt Ellmann, Jim Kupsch, Shelly Guo, Johan Larson, David De Witt, and Jeffrey Naughton. 1997. Building A Scaleable Geo-Spatial DBMS: Technology, Implementation and Evaluation. *SIGMOD Rec.* 26, 2 (1997), 336–347. DOI: <http://dx.doi.org/10.1145/253262.253342>
- A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. 2009. A Comparison of Approaches to Large-Scale Data Analysis. (2009).
- Alexei Pozdnoukhov and Christian Kaiser. 2011. Scalable Local Regression for Spatial Analytics. (2011). DOI: <http://dx.doi.org/10.1145/2093973.2094023>
- Dimitrios Pountzos and Keshav Pingali. 2013. Betweenness centrality: algorithms and implementations. *SIGPLAN Not.* 48, 8 (2013), 35–46. DOI: <http://dx.doi.org/10.1145/2517327.2442521>
- G. Qin and L. Gao. 2012. An algorithm for network motif discovery in biological networks. *International Journal of Data Mining and Bioinformatics* 6, 1 (2012), 1–16.
- Judy Qiu, Jaliya Ekanayake, Thilina Gunarathne, Jong Youl Choi, Seung-Hee Bae, Hui Li, Bingjing Zhang, Tak-Lon Wu, Yang Ryan, Saliya Ekanayake, Adam Hughes, and Geoffrey Fox. 2010. Hybrid cloud and cluster computing paradigms for life science applications. *BMC Bioinformatics Proceedings of BOSC 2010* (2010). <http://grids.ucs.indiana.edu/ptliupages/publications/HybridCloudandClusterComputingParadigmsforLifeScienceApplications.Pub.pdf>
- Judy Qiu, Jaliya Ekanayake, Thilina Gunarathne, Jong Youl Choi, Seung-Hee Bae, Yang Ruan, Saliya Ekanayake, Stephen Wu, Scott Beason, Geoffrey Fox, Mina Rho, and Haixu Tang. 2011. *Data Intensive Computing for Bioinformatics*. IGI Publishers. DOI: <http://dx.doi.org/10.4018/978-1-6152971-2>
- Judy Qiu and Bingjing Zhang. 2013. *Clustering Social Images with MapReduce and High Performance Collective Communication*. IOS Press. <http://grids.ucs.indiana>

- edu/ptliupages/publications/MammothDataintheCloudClusteringSocialImage.pdf
- Judy Qiu and Bingjing Zhang. 2014. Harp: a runtime for efficient in-memory communication. (2014). <http://salsaproj.indiana.edu/harp/>
- R Project. 2012. R open source statistical library. <http://www.r-project.org/>. (2012).
- P. Ribeiro, F. Silva, and L. Lopes. 2012. Parallel discovery of network motifs. *J. Parallel and Distrib. Comput.* 72, 2 (2012), 144–154.
- Ken Rose. 1998. Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems. *Proc. IEEE* 86 (1998), 2210–2239.
- Ken Rose, Eitan Gurewitz, and Geoffrey Fox. 1990. A deterministic annealing approach to clustering. *Pattern Recogn. Lett.* 11 (1990), 589–594.
- Yang Ruan, Saliya Ekanayake, Mina Rho, Haixu Tang, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. 2012a. DACIDR: Deterministic Annealed Clustering with Interpolative Dimension Reduction using Large Collection of 16S rRNA Sequences. (October 7-10 2012). http://grids.ucs.indiana.edu/ptliupages/publications/DACIDR_camera_ready_v0.3.pdf
- Yang Ruan and Geoffrey Fox. 2013. A Robust and Scalable Solution for Interpolative Multidimensional Scaling with Weighting. (October 22-25 2013). DOI: <http://dx.doi.org/10.1109/eScience.2013.30>
- Yang Ruan, Zhenhua Guo, Yuduo Zhou, Judy Qiu, and Geoffrey Fox. 2012b. HyMR: a Hybrid MapReduce Workflow System. (June 18 2012). http://grids.ucs.indiana.edu/ptliupages/publications/HyMR_submission_HPDC_workshop_final.pdf
- Yang Ruan, Geoffrey L. House, Saliya Ekanayake, Ursel Schtte, James D. Bever, Haixu Tang, and Geoffrey Fox. 2014. Integration of Clustering and Multidimensional Scaling to Determine Phylogenetic Trees as Spherical Phylograms Visualized in 3 Dimensions. (May 26-29 2014). <http://grids.ucs.indiana.edu/ptliupages/publications/PhylogeneticTreeDisplayWithClustering.pdf>
- S. Sarkar and A. Dong. 2011. Community detection in graphs using singular value decomposition. *Physical Review E* 83, 4 (2011), 04611.
- V. Satuluri and S. Parthasarathy. 2009. Scalable graph clustering using stochastic flows: applications to community discovery. (2009).
- Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. 2011. Local graph sparsification for scalable clustering. (2011). DOI: <http://dx.doi.org/10.1145/1989323.1989399>
- D. A. Schiffmann, D. Dikovskaya, P. L. Appleton, I. P. Newton, D. A. Creager, C. Allan, I. S. Nathke, and I. G. Goldberg. 2006. Open microscopy environment and findspots: integrating image informatics with quantitative multidimensional image analysis. *Biotechniques* 41, 2 (2006), 199–208. DOI: [http://dx.doi.org/000112224\[pil\]](http://dx.doi.org/000112224[pil])
- C. Seshadhri, T. Kolda, and A. Pinar. 2012a. Community structure and scale-free collections of Erdos–Renyi graphs. *Physical Review E* (2012). <http://arxiv.org/abs/1112.3644>
- C. Seshadhri, A. Pinar, and T. Kolda. 2012b. Fast Triangle Counting through Wedge Sampling. (2012). <http://arxiv.org/abs/1202.5230>
- Larissa Stanberry, Roger Higdon, Winston Haynes, Natali Kolker, William Broomall, Saliya Ekanayake, Yang Ruan, Judy Qiu, Eugene Kolker, Geoffrey Fox, and Adam Hughes. 2012. Visualizing the Protein Sequence Universe. (June 18 2012). http://grids.ucs.indiana.edu/ptliupages/publications/paperDelft_final.pdf
- Antonio Torralba, Kevin P Murphy, William T Freeman, and Mark A Rubin. Context-based vision system for place and object recognition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 273–280.
- Charalampos E. Tsourakakis, U Kang, Gary L. Miller, and Christos Faloutsos. 2009. DOULION: Counting triangles in massive graphs with coin. (2009).

- Stanford University. 2014. ImageNet image database organized according to the WordNet hierarchy. (2014). <http://www.image-net.org/>
- Fusheng Wang, Jun Kong, Jingjing Gao, Lee A.D. Cooper, Tahsin Kurc, Zhengwen Zhou, David Adler, Cristobal Vergara-Niedermayr, Bryan Katigbak, Daniel J Brat, and Joel H Saltz. 2013. A high-performance spatial database based approach for pathology imaging algorithm evaluation. *J Pathol Inform.* 4, 5 (2013).
- F Wang, T Kurc, P Widener, T Pan, J Kong, L Cooper, D Gutman, A Sharma, S Cholleti, V Kumar, and J Saltz. 2010. High-performance Systems for In Silico Microscopy Imaging Studies. *The 7th International Conference on Data Integration in the Life Sciences, Gothenburg, Sweden* (2010).
- Kaibo Wang, Yin Huai, Rubao Lee, Fusheng Wang, Xiaodong Zhang, and Joel Saltz. 2012. Accelerating Pathology Image Data Cross Comparison on CPU-GPU Hybrid Systems. *Proc. VLDB Endow.* 5, 11 (2012), 1543–1554.
- Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on.* IEEE, 3485–3492.
- Shubin Zhang, Jizhong Han, Zhiyong Liu, Kai Wang, and Shengzhong Feng. Spatial Queries Evaluation with MapReduce. In *Eighth International Conference on Grid and Cooperative Computing.* 287–292. DOI: <http://dx.doi.org/10.1109/gcc.2009.16>
- Shubin Zhang, Jizhong Han, Zhiyong Liu, Kai Wang, and Zhiyong Xu. SJMR: Parallelizing Spatial Join with MapReduce on Clusters. In *IEEE International Conference on Cluster Computing.* 1–8. DOI: <http://dx.doi.org/10.1109/clustr.2009.5289178>
- Y. Zhang, Z. Wang, Y. Wang, and L. Zhou. 2009. Parallel community detection on large networks with propinquity dynamics. (2009).
- Zhao Zhao, Maleq Khan, V. S. Anil Kumar, and Madhav V. Marathe. 2010. Subgraph Enumeration in Large Social Contact Networks Using Parallel Color Coding and Streaming. (2010). DOI: <http://dx.doi.org/10.1109/icpp.2010.67>
- Z. Zhao, G. Wang, A. Butt, M. Khan, V. S. Anil Kumar, and M. Marathe. 2012. SAHAD: Subgraph Analysis in Massive Networks Using Hadoop. (May 2012).